A Software Process Ontology and Its Application

Li Liao[1], Yuzhong Qu[1], Hareton K. N. Leung[2]

[1] Department of Computer Science and Engineering, Southeast University, Nanjing, 210096, P. R. China
{xobjects, yzqu}@seu.edu.cn
[2] Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong
cshleung@comp.polyu.edu.hk

Abstract. Software process is viewed as an important factor to deliver high quality products. Although there have been several Software Process Models proposed, the software processes are still short of formal descriptions. This paper presents an ontology-based approach to express software processes at the conceptual level. An OWL-based ontology for software processes, called SPO (Software Process Ontology), is designed, and it is extended to generate ontologies for specific process models, such as CMMI and ISO/IEC 15504. A prototype of a web-based process assessment tool based on SPO is developed to illustrate the advantages of this approach. Finally, some further research in this direction is outlined.

1 Introduction

A software process is defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products [1]. It is viewed as a vehicle to improve software quality as well as productivity. A number of software process (SP) models have been developed, such as the Capability Maturity Model (CMM) and ISO/IEC 15504.

The CMM was developed by SEI (Software Engineering Institute) in early 1990s [1]. As a de facto standard for process assessment and improvement, CMM is used to identify the key elements of effective software processes and to evaluate the maturity of software processes of an organization. The ISO/IEC 15504 model is an international standard for software process assessment [2]. It was first published in 1998 as a Technical Report Type 2. It consists of two dimensions. The process dimension provides the definition of the processes, and the capability dimension describes a series of process attributes, which represent the measurable characteristics of the processes. Besides these two models, there are other assessment models such as BOOTSTRAP [3], Trillium [4], and CMMI [5], etc. With these models, software organizations can assess and improve their processes to become more competitive and produce high quality products. However, using these models is difficult for most

software organizations because they are various in international or regional characteristics, purposes, orientation and structures. Some of the problems emerged in the usage of the Process Models are as following [4]:

- Formal description of process models

Almost all existing process models are empirical and descriptive models. They lack rigorous and formal description of model structure and process framework. Software organizations often collect data and assess their processes by checklists. Problems of ambiguity, instability, too much subjectivity, and inaccuracy in process assessment and application were identified in existing process models. The positive effects of the models are not very evident [6].

Ontology can eliminate conceptual and terminological confusion, and provides a representation vocabulary specialized to the software processes. Unified terms and conceptions of ontologies enable knowledge sharing, and ontological analysis clarifies the structure of knowledge [14]. If we can use ontology to represent processes and process models, then we could provide formal descriptions to them, and use reasoning functions to assist in the analysis of the models.

- Compatibility & Transformability

Compatibility and transformability between models are fundamental requirements for the software organizations. The current process models exhibit different orientations in software process modeling. Divisions between current process models cause many problems in comparative analysis and modeling, and the compatibility of the current process models and their assessment results are found to be limited [7].

By creating ontologies for the current process models and using ontology alignment techniques, we can solve the compatibility problem without the cost of changing the existing models. And by relating the assessment results of process capability levels between different process models, a software organization can also avoid multiple and costly assessments.

- Benchmark of process attributes

Quantitative analysis and benchmark of process attributes are other foundations needed to validate a model at a lower level. There are few reports of benchmarks for the current models to be found in literature, because collecting data by questionnaires is very difficult. With ontology and semantic web, collecting data on the Internet and developing benchmarks of software processes in some areas would become easier.

In this paper, we present an ontology-based approach to express software processes at the conceptual level. We designed a software process ontology, called SPO, which defined the structure of the process models at the schema level. SPO is semantic rich, reusable, and has good extensibility. In section 2, we discuss previous work on software process representation and related tools. In section 3, we abstract a common architecture of software process model from existing models, and illustrate a general overview of SPO and then extend it to suit the needs of CMMI and ISO/IEC 15504. In section 4, we discuss the usage of the SPO as well as its extensions. Finally, we conclude the paper and identify some future research directions.

2 Related work

Even before the appearance of these SPI models, people have identified that the software processes should be unambiguous and the relationship among the processes should be considered. People have investigated on how to describe software processes precisely with Knowledge Representation techniques, and some related Process-centered Software Engineering Environments (PSEEs) have been developed, such as EPOS [8], Marvel [9], SPADE [10], DPSSEE [11], IDERS [12], etc.

With Knowledge Representation techniques, these environments added logic rules to the processes, so that they could provide appropriate management and utilization of the information, and speed up the software development process. But most of the existing PSEEs could not meet the requirements of the software organizations, because they only focus on the lifecycle models, oriented to the development processes. Their coverage is limit. And till now, no Process Modeling language (PML) or PSEE supports the existed software process models.

There are also some SPI tools that can help to improve the processes by providing many functions such as process assessment, problem analysis, change management, document management, etc. Our previous study[13] found 38 such tools.

Most of the tools are based on one of the popular process models, and provide process assessment. Their data models lack flexibility, so their extensibility is limited. No tool supports the mapping between the models. Another problem is that most tools store the data in their own format. This affects the interoperability of the tools, and causes difficulty in integrating these tools together to provide all the needed SPI functions to the organizations.

Because of the drawbacks discussed above, the usage of these tools and environments is limited. A formal description of the process models and an integrated SPI environment that is extensible, effective, inexpensive, and easy to use, are urgently needed.

3 Software Process Ontology (SPO)

In this section, we abstract a common architecture of software process model from existing popular models, and design a unified set of atomic practices as a main component of our software process ontology. And then we illustrate a general overview of SPO and extend it to suit the needs of CMMI and ISO/IEC 15504.

3.1 Abstract architecture of Software Process Model

Although the architectures of the software process models are different and their model components have various names, they nevertheless have some similarities [4]. Their main components are “Process” and “Practice”. Normally, the models have a set of processes, which could guide the software production, and the processes are classified into several domains, called “Subsystem” or “Category”. The organizations

must carry out practices to reach the goals of the processes. Taxonomies of the model components for several Process Models are shown in Table 1 [4].

Table 1. The taxonomies of the models' components

Components / Model	Subsystem	Category	Process	Sub-Process	Practice	Process Attribute
CMM		Category	Key Process Area		Key Practice	
CMMI		Category	Process Area	Specific Goal	Specific Practice	Generic Goal
ISO/IEC 15504		Category	Process	Component Process	Base Practice	Process Attribute
ISO 9001	Subsystem		Main topic area		Management issue	
BOOTSTRAP	Process Area	Process Category	Process		Practice	

From Table 1, we can conclude that the "Category/ subsystem", "Process" and "Practice" are the common components of the models. The CMMI and ISO/IEC 15504 are two-dimension models, so they have a special component, "Process Attribute", to evaluate the mature level of the processes.

From our investigation we found that not only the structures of the models are similar, but also the coverage of these models overlaps. We use the CMMI continuous model and the ISO/IEC 15504 model to discuss this issue. We choose these two models because they are the most popular software process models in the world, and they affect each other during their development.

In our investigation, we found the scopes of the two models are largely overlapped. Although their process names are not the same, their contents are similar. Supposing that two processes could be considered mapped when more than 75% of their contents are the same, we obtain Table 2, which illustrated mappings between the processes of the two models.

Table 2. The comparison between CMMI and ISO/IEC 15504 processes

Process Areas (CMMI)	Process (ISO/IEC 15504)
Supplier Agreement Management	Acquisition
Supplier Agreement Management	Supply
	Operation
Requirement Management	Development

Requirement Development	Requirement Elicitation / Development
Technical Solution	Development
Product Integration	Development
Verification	Development/ Verification
Validation	Development/ Validation
	System and software maintenance
Configuration Management	Configuration Management
Process and Product Quality Assurance	Quality Assurance
Measurement and Analysis	
Decision Analysis and Resolution	Problem resolution
Causal Analysis and Resolution	Problem resolution
Organizational Environment for Integration	
	(Part of)Verification
	(Part of)Validation
	Documentation
	Joint review
	Audit
Project Planning	Project Management
Project Monitoring and Control	Project Management
	Quality Management
Integrated Project Management for IPPD	Project Management
Risk Management	Risk Management
Integrated Teaming	Organizational alignment
Integrated Supplier Management	
Quantitative Project Management	Measurement
Organizational Process Focus	Process Management
Organizational Process Definition	Infrastructure/ Process Management
Organizational Training	Human resource management
Organizational Process Performance	Process Management
Organizational Innovation and Deployment	Improvement process /Reuse

From Table 2, we can find out that the two models have different processes, but the contents of these processes can be mapped. Each model has its own concepts and

terms, and even processes that have the same name in the two models usually have different contents. Same findings apply to the practices from these models. For example, the Specific Practice “SP 1.2-1 Appraise the Organization’s Processes” in CMMI has similar content as the Process “process assessment sub process” in ISO/IEC 15504.

3.2 Atomic practice model

As discussed in section 3.1, the processes of the models are partly overlapped, and the granularity of the processes and practices is different in the models. Trying to solve this problem and implement the mapping between the models, we collect all the activities available for the models, called Atomic Practices, and construct a unified set of atomic practices, called Atomic Practice Model (APM). The practices and processes of the models can be composed from the APM. APM is a main component of our software process ontology.

The atomic practice is the minimal activity that can develop software artifacts or support the engineering process. A software process is composed of a collection of practices, and a practice comprises a collection of the atomic practices. For example, “Capture all requirements and requirements changes” is an activity of ISO15504 practice CUS.3.BP4, at the same time it is also an activity of CMMI practice REQM.SP.1.3-1. So we define this activity as an atomic practice. An atomic practice can include the following attributes:

- Activity Name and Purpose
- Artifacts used/ required
- Task description
- Task responsibility
- Product(s)/Document(s) developed
- Measures

Till now, we have constructed a sample APM based on some of the activities in CMMI SE/SW/IPPD/SS 1.1 continuous model and ISO/IEC 15504. In current version of SPO, we consider the activity name, task description and measure of atomic practice, and more attributes will be incuded in next version of SPO.

3.3 Framework of SPO

With the capability of OWL [15], we designed an ontology-based software process model framework, Software Process Ontology, which defined the process model at the schema level.

In SPO, we defined some classes to represent components in models, and properties to represent the relationship between components. The Resource Description Framework (RDF) Graph of SPO is shown in Figure 1.

The hierarchies between the classes represent that a class is a sub-class of another class. For example, “Process” is a class representing the super class of all kinds of the processes in the models; it has two sub-classes, “CompositeProcess” and “BasicProcess”. The former represents the super class of the Process Areas in CMMI

or the Processes in ISO/IEC 15504, and the latter represents the super class of the Specific Goals in each Process Area or the Sub-Processes in each Process. Similarly, we designed the “Practice” class as a super class of all the practices in the models; its subclass, “BasicPractice” represents the super class of the Practices in the models, “AtomicPractice” represents the atomic practices in APM, and its subclasses, from AP1 to APn, are the contents(atomic practices) of APM that could buildup the practices.

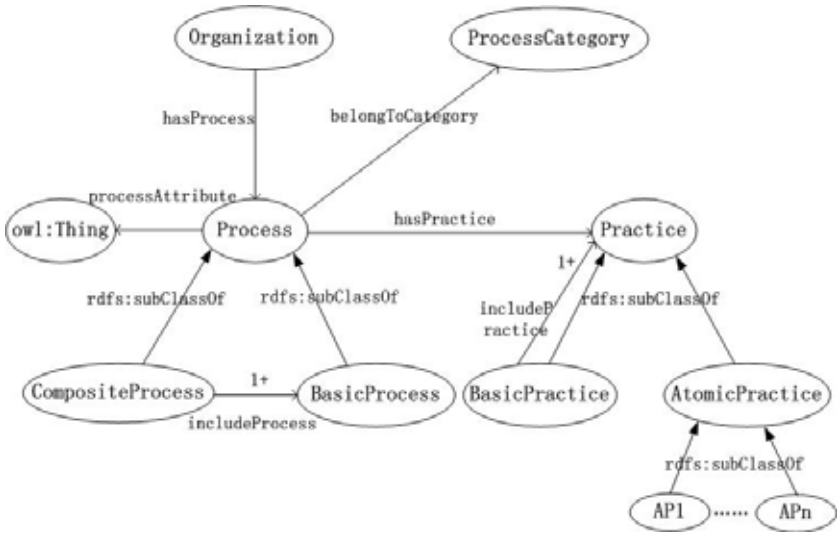

Fig. 1. RDF Graph of SPO

Besides those classes, there are classes named “SoftwareOrganization”, “ProcessCategory” as well. The “ProcessCategory” class represents the process category in the models, which is an important component representing the domains of the processes. The “SoftwareOrganization” is the class representing the organization, with the real organizations being its instances in the usage of SPO.

The relationships between these concepts are explicitly represented by properties. For example, the property “hasPractice” has a domain of “Process” and a range of “Practice”. Using this we can express that a process has one or more practices. That is to say if an organization adopt a particular process, it must implement its practices.

There are some other properties besides those represented in the graph. For example, the class “Process” has some properties similar to Atomic Practice. Those properties also describe the purpose, requirements, responsibilities and productions of a process. In addition to these properties, the class “Process” can have two more properties, one is “preProcess”, which shows the processes that should have finished before a process begins, and the other is “nextProcess”, which suggests the processes that may do after a process has finished. These two properties are useful for constructing the workflow of organizations’ processes, and we plan to add them to SPO in the next version.

3.4 Extension of SPO

Based on SPO, we build CMMI_Onto and ISO15504_Onto separately. They are two extensions of the SPO designed to fit with the CMMI model and the ISO/IEC 15504 model, respectively. As discussed above, the Process Areas, Specific Goals, Specific Practices in CMMI are defined respectively as the sub-classes of "CompositeProcess", "BasicProcess", "BasicPractice". This represents that a Process Area has one or more Specific Goals, and the organizations must finish related practices to achieve a goal.

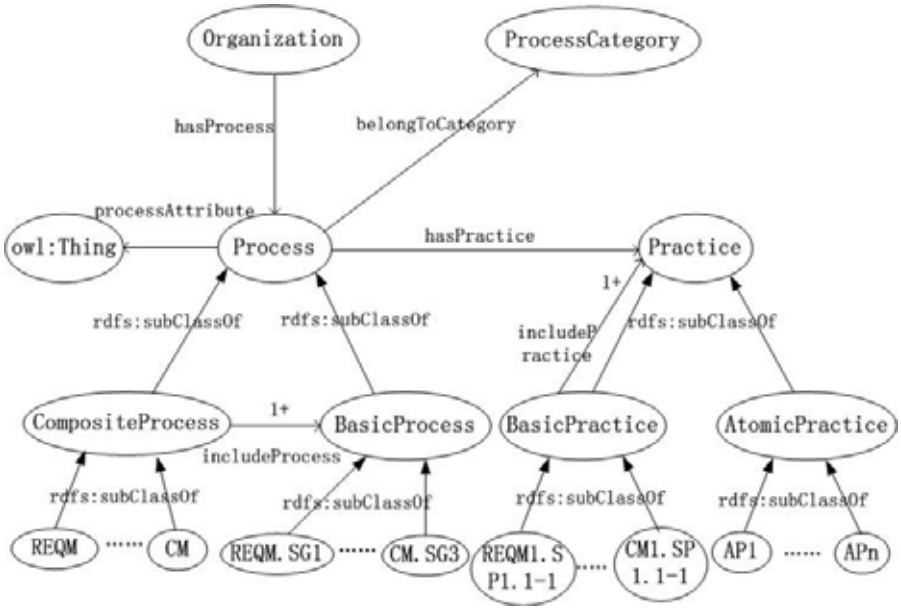

Fig. 2. RDF Graph of CMMI_Onto

For example, in Figure 2, the RDF Graph of CMMI_Onto, we can see that REQM (Requirements Management Process Area) is a subclass of "CompositeProcess", and REQM.SG1 (Requirements Management Process Area) is a subclass of "BasicProcess". Then we can use the relational property "includeProcess" to specify that "REQM.SG1" is one of the Specific Goals of "REQM". Similarly, we can use other properties, such as "hasPractice" "includePractice", to specify the relationship between the processes and practices, and the relationship between the practices and atomic practices.

The ISO15504_Onto is constructed in the same way. Because both models are based on SPO, and composed of atomic practices, CMMI_Onto and ISO15504_Onto can be used to map the two Process Models.

4 Usage of SPO and its extensions

Under the ontology groundwork, we designed a prototype of a web-based process assessment tool[1] to prove our idea. When a user logs on our assessment web site, s/he can select a reference model and the category of process that s/he is concerned with. Then, the user can select the processes her/his organization adopted. After that, the user will get the description of processes, sub-processes, practices and atomic practices that the reference model recommended. The user can perform appraisal of the processes and practices, and then store the evaluation results.

Our tool has not been finished yet, but it has demonstrated that different reference models can be involved in one tool. It is hopeful that ontology could help to solve the issues we discussed earlier. Using the mechanism of inheritance and restriction in OWL, it will be easy to extend the core ontologies to match the revision of the reference models. All the reference models are composed by the contents of APM. That improved the transferability among the models. Further, as the collected data are stored in XML format, they could be reused by other SPI tools and also could be used for the benchmark of the processes in the future.

Our ontologies can also help to construct process model for the software organizations. Because in SPO, we have defined the atomic practice model, so the organizations can define their own process by adopting related atomic practices, according to their own conditions and status. We also have ontologies for two existing reference models, which are composed on the same atomic practice model, so the organizations can also map their own model to these reference models.

5 Conclusion and Future work

We have presented SPO, an OWL-based ontology for software processes. We also briefly presented how to extend SPO to CMMI_Onto and ISO15504_Onto, which fit with CMMI model and ISO/IEC 15504 model respectively. Using these ontologies, we implemented a prototype web-based process assessment tool.

This is just a start towards semantic description for software processes. The next step of our study involves consummating our process ontology, developing a matching algorithm for the mapping between the models, and extending the functions of the tool. We are also considering how to use the reasoning capability of ontologies to help the organizations to construct their own process models. After revising the tool, we will do some empirical studies, in order to improve the tool. With the data collected by the assessment tool, we will be able to develop a benchmark of software processes in some areas.

[1] http://cse.seu.edu.cn:8080/spo/index.jsp

Acknowledgments

This work is supported in part by National Key Basic Research and Development Program of China under Grant 2003CB317004, in part by Hwa-Ying Culture and Education Foundation, and in part by JSNSF under Grant BK2003001. We would like to thank Ningsheng Jian and Dongdong Zheng in our team for their contributions in the prototype implementation. We would also like to thank anonymous reviewers for their suggestions on this paper.

References

1. Paulk M.C., Curtis, B., Chrissis, M.B., Weber, C.V.(eds.):CMM Capability Maturity ModelSM for Software. Version 1.1, Technical Report, CMU/SEI (1993)
2. ISO/IEC 15504 Standard For Software Process Assessment (Parts 1-9), ISO/IEC Technical Report (TR), International Standards organization (1998~2004)
3. Haase, V., Messnarz, R., Koch, G.,et al: Bootstrap: Fine-Tuning Process Assessment. IEEE Software (1994) 25-35
4. Wang, Y.X., King, G.: Software Engineering Processes- Principles and Applications. CRC Press LLC (2000)
5. CMU/SEI: Capability Maturity Model® Integration (CMMISM), Version 1.1: CMMISM for Software Engineering (CMMI-SW/SE/IPPD/SS, V1.1) Continuous Representation. (2002)
6. Conradi, R., Fuggetta, A.: Improving software process improvement. IEEE Software, (2002) 92-99
7. Lopez, M.: the Software Process: Evaluation and Improvement, in Proceedings of World Multi conference on Systemics. Cybernetics and Informatics, ISAS-SCIs 2001, Vol. I, Orlando, Florida (2001) 255-260
8. Conradi R., Hagaseth M., Larsen J.O., et al.: EPOS: Object-oriented cooperative process modeling, Software Process Modeling and Technology (1994) 33-70
9. Ben-Shaul, I.Z., Kaiser, G.E.: Process Evolution in the Marvel Environment. in Proceedings of 8th International Software Process Workshop: State of the Practice in Process Technology, Wadern, Germany (1993) 104-106
10. Bandinelli, S., Braga, M., Fuggetta, A., Lavazza, L.: Cooperation support in the SPADE Environment: A Case Study, in Proceeding of Workshop on Computer Supported Cooperative Work. Petri Nets and Related Formalisms, Chicago (1993)
11. Deng, D., Wang, T., Sheu, P.C.-Y., Maezawa, H., Tsunoda, F., Onoma, A.K.: DPSSEE: a distributed proactive semantic software engineering environment. Multimedia Software Engineering, 2003, in proceedings of the Fifth International Symposium (2003) 124-131
12. Alonso, A., Christensen, H., Baresi, L., Heikkinen, M.: IDERS: an integrated environment for the development of hard real-time systems. Real-Time Systems, 1995, in Proceedings of Seventh Euromicro Workshop (1995) 4-10
13. Liao, L., Leung, H.K.N., Qu, Y.Z.: Automated Support of Quality Improvement. In proceedings of The 8th IASTED International Conference on SOFTWARE ENGINEERING AND APPLICATIONS (2004) 270-276
14. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: What Are Ontologies, and Why Do we Need Them?. IEEE Intelligent Systems Vol. 14, Issue 1 (1999) 20-26.
15. Patel-Schneider, P.F., Hayes, P., Horrocks, I. (eds.): OWL: Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation 10 February 2004. Latest version is available at http://www.w3.org/TR/owl-semantics/